university of groningen / faculty of science and engineering / artificial intelligence

Sparse Rewards Reinforcement Learning: Addressing Vanishing Intrinsic Rewards in Change-Based Exploration Transfer

Bachelor's Project Thesis

Diana-Maria Arapu, s4729722, d.arapu@student.rug.nl,
Supervisor: R. Fernandes Cunha

Abstract: Exploration is one of the main challenges in sparse rewards reinforcement learning. The Change-Based Exploration Transfer (C-Bet) method leverages intrinsic motivation to learn an exploration policy that mitigates the lack of external rewards and guides exploration. The agent receives a higher intrinsic reward when visiting less frequented areas and while performing actions that change the environment. However, because these intrinsic rewards are solely relied upon to learn the exploration policy, they diminish over time and eventually vanish. This study proposes a novel approach to address this issue by introducing a time-variant component to the C-Bet intrinsic reward. The modified algorithm was evaluated in various MiniGrid environments, procedurally-generated gridworlds that present exploration challenges due to sparse rewards. The results indicate that the modified algorithm improves exploration in some environments compared to the original C-Bet algorithm. However, its performance depends on the structure of the environment, showing limited generalization. Nonetheless, this improvement highlights the potential for more effective exploration strategies that rely on intrinsic motivation.

1 Introduction

Exploration is one of the main challenges in reinforcement learning (RL) as it requires building an agent that behaves human-like when interacting with an environment (Parisi et al., 2021). This may be difficult, especially regarding real-world problems, where rewards are sparse or nonexistent. Several exploration-based algorithms inspired by human learning have been employed to solve tasks in such settings. Human behavior is influenced by two distinguished processes, namely extrinsic and intrinsic motivation (Ryan and Deci, 2000). While extrinsic motivation refers to acting in a certain way due to an externally provided reward, intrinsic motivation denotes "doing something because it is inherently interesting" (Dohare et al., 2023). In the context of RL, extrinsic motivation is represented by well-defined rewards received by the agent as it is approaching a specified goal, therefore engaging in task-driven exploration (Parisi et al., 2021). However, such rewards may be lacking in sparse reward environments. One way to solve this issue is to use intrinsic motivation to encourage exploration in a task-agnostic manner (Barto, 2012). The first uses of intrinsic rewards in RL involved implementing forms of curiosity or exploration bonuses (Schmidhuber, 1991; Sutton, 1991). In later applications, the intrinsic reward may be characterized based on two components of exploration: an agent-centric component and an environment-centric component (Parisi et al., 2021).

Agent-centric exploration relies on the agent's own beliefs to encourage the visitation of unseen and surprising states in the environment. Count-based exploration methods employ an agent-centric component, namely the visit counts, guiding the agent towards less visited state-action pairs (Tang et al., 2017). However, Bellemare et al. argue that visit counts are ineffective in practical settings, as states are hardly ever revisited (2016). They propose an alternative, the Count algorithm, which introduces an intrinsic reward based on pseudo-counts. Pseudo-counts are derived from a density model over the state space, which outputs proba-

bility distributions representing the likelihood of a state to be visited. When a state is encountered, the density model's prediction for that state increases, corresponding to a unit increase in pseudo-count.

On the other hand, environment-centric exploration directs the agent towards inherently interesting states within the environment. Emphasizing changes in the environment rather than visitation counts may aid the agent in learning a policy that effectively generalizes across an extensive state space. This approach is particularly useful in procedurally-generated environments (PGEs), where the agent faces the same task but encounters a different environment layout in each episode(Raileanu and Rocktäschel, 2020).

While these algorithms aim to mitigate the absence of extrinsic rewards, Parisi et al. argue that the commonly-used task-agnostic exploration setup is impractical and inconsistent with human-like exploration (2021). This approach assumes that agents do not utilize experience from previous environments when exploring new ones. In contrast, humans use their prior knowledge to develop new and improved strategies. The authors propose a novel algorithm, Change-Based Exploration Transfer (C-Bet), which employs a more realistic setup composed of two phases. In the first phase, the exploration phase, the agent interacts with one or many environments without an extrinsic goal, thus engaging in task-agnostic exploration and learning an exploration policy based on intrinsic rewards. In the second phase, the agent explores new, unseen environments in a task-specific manner, using the knowledge transferred from the exploration policy along with extrinsic rewards. This setup considers exploration from a continual lens, as the agent's experience is retained across multiple environments. Moreover, the authors introduce a new intrinsic reward mechanism that includes both an agent-centric and an environment-centric component. The agent-centric component is the visitation state count, which encourages the exploration of unseen states, similar to Count-based algorithms. The environment-centric component is the environment change count. Specifically, when in a certain state, the agent may choose an action that alters the environment, which can be identified by comparing the next state to the current state. The authors emphasize that rare changes are inherently interesting, so actions leading to such changes should be favored, while actions that don't affect the environment should be penalized (Parisi et al., 2021).

This intrinsic reward presents an issue when utilized independently during the exploration phase. As the agent explores, the counts grow, causing the intrinsic rewards to decrease until they vanish, providing no further guidance to the agent. Parisi et al. propose a solution, to reset the counts with a small probability. Using random resets instead of episodic resets ensures that the initial states are not always favored.

While this implementation has effectively resolved the problem of vanishing rewards, there may be more ways to tackle this issue. For instance, multiplying the intrinsic reward by a specific factor could significantly slow down its gradual decrease over time. Therefore, it would be beneficial for this factor to be a dynamic function that adjusts over time. Introducing a time-varying function into the intrinsic reward may solve the problem of vanishing rewards in the exploration phase. Furthermore, the counts do not have to be reset. This allows the agent to keep track of all visited trajectories, mimicking how humans rely on their past experience to benefit exploration.

The primary goal of this study is to assess how scaling up the intrinsic reward function by a time-variant component affects the performance and speed of convergence of C-Bet with no count resets. Hypothetically, the modified algorithm would improve exploration and outperform the original C-Bet method, showing a higher extrinsic return at transfer to new environments. To evaluate the modified algorithm's performance, we compare it to the original C-Bet algorithm with count resets, as well as the Count algorithm, which demonstrated a comparable performance in the original study. Additionally, we include two environments not previously used in C-Bet experiments. These environments are particularly interesting because taking actions that produce changes in the environment may be harmful in these settings. This challenges the intuition behind the advantage of considering environment change counts in the intrinsic reward to enhance exploration.(Nahirnyi, 2022).

2 Methods

2.1 Proposed approach

The intrinsic reward used in the C-Bet algorithm, denoted by r_i, is defined as such:

$$r_i(s, a, s') = \frac{1}{N(s') + N(c)}, \tag{2.1}$$

where $c(s, s')$ is the environment change given a transition (s, a, s') and N is the count of changes and states. A change in the environment is represented as the difference between 360° panoramic views. These are preferred as they are regarded as rotation-invariant representations of the states, designed to represent the environment changes rather than the agent's state.

The first step in modifying this reward function is deciding on a time-variant function. We considered two possible functions: a linear function t or a logarithmic function $ln(t)$. To decide between these two functions, we considered the shape of the denominator of the intrinsic reward, $N(s')+N(c)$. Intuitively, in the limit, both functions are linear, as each count increases by a constant value. However, taken at an arbitrary timestep, they are not linear, as the agent may visit the same state or change after different periods of time. Therefore it would be more appropriate to use $ln(t)$ in this reward function. Taking this into consideration, the Modified C-Bet's intrinsic reward, denoted by $\hat{r_i}$, is defined as such:

$$\hat{r_i}(s, a, s') = \frac{1 + ln(t)}{N(s') + N(c)}, \tag{2.2}$$

where $t >= 1$ represents the time-step since the beginning of training.

This approach is based on the assumption that the intrinsic rewards in Eq. 2.1 decrease over time to the point that they reach zero given enough samples. The proposed modification in Eq. 2.2 aims to significantly slow down this decrease to the point where this would allow for any realistic training setup to be executed without having the issue of vanishing rewards. Moreover, this would be done without resetting the state count or the change count functions. Hypothetically, keeping track of all counts in the pre-training phase would allow the agent to rely on all past experiences, which may benefit exploration. Moreover, a good exploration policy may improve the performance of the task-specific policy at transfer to new environments

2.2 Architectural details

The code is built on the C-Bet implementation, which is open-sourced and available at `https://github.com/sparisi/cbet/`. The C-Bet algorithm uses IMPALA (Espeholt et al., 2018), an off-policy actor-critic RL method, both in the pre-training phase to learn the exploration policy π_{EXP} and in the transfer phase to learn the task-specific policy π_{TASK}. The exploration policy is of the form $\pi_{EXP}(s, a) = \sigma(f_i(s, a))$, where σ is the softmax function and f_i a function representing the value of states $V(s)$ trained on the intrinsic rewards, suggested by the subscript "i". The task-specific policy has a similar form, but it additionally integrates f_i as a bias: $\pi_{TASK}(s, a) = \sigma(f_e(s, a) + f_i(s, a))$, where f_e is trained using extrinsic rewards r_e, indicated by the subscript "e". IMPALA distinguishes itself from other actor-critic methods by employing multiple actors that operate in parallel, optimising resource utilisation. The actors follow a behaviour policy μ to generate trajectories of experience, which are then fed to a central learner responsible for learning the value function V^π under the target policy π. Given that the behaviour and target policies differ, this setup corresponds to an off-policy learning framework. At the outset of each trajectory, each actor updates its policy to match the most recent learner policy and interacts with the environment over several steps. However, the learner's policy may be way ahead of the actor's policy, which causes a "policy-lag" between the actors and learners. Espeholt et al. implemented the V-trace algorithm to address this issue. This algorithm computes the V-trace targets, which use truncated importance sampling ratios to correct for the difference between the two policies. The importance sampling ratio is given by:

$$\rho_t = \frac{\pi(a_t|s_t)}{\mu(a_t|s_t)}, \tag{2.3}$$

where π represents the target policy and μ denotes the behaviour policy. The ratios are truncated using a predetermined threshold to prevent them from becoming excessively large and potentially destabilizing the training process.

The modified C-Bet algorithm shares a nearly identical implementation with the original, with a few minor differences. Firstly, the intrinsic reward is defined by Eq. 2.2. Secondly, the counts in the intrinsic reward feature a count reset probability of 0. Thirdly, the intrinsic reward coefficient has a different value. This coefficient is used to scale down the rewards for numerical stability. In the original C-Bet, this coefficient is set to 0.005. Running the modified C-Bet with this coefficient shows a big discrepancy in the intrinsic rewards compared to C-Bet, notably higher due to scaling by the time-variant component.To maintain a similar reward range as the original algorithm, the coefficient for the modified C-Bet is adjusted to 0.000625. This specific value was determined by analyzing the intrinsic rewards of both algorithms when trained in one of the environments (see Appendix A).

2.3 Environments

MiniGrid is a library featuring procedurally-generated environments (PGE) where agents solve various tasks by interacting with objects in a 2D gridworld (Chevalier-Boisvert et al., 2023). Each environment includes objects like keys, doors, and boxes. Exploration in these environments is challenging due to the varied layout of each PGE upon reset and sparse rewards. Specifically, agents only receive rewards upon reaching the goal, based on the number of steps taken. Figure 2.1 illustrates the environments used in this study, corresponding to the environments proposed by Parisi et al., with two additions: LavaCrossing and DynamicObstacles. These environments are included because actions that induce changes may lead to detrimental outcomes.

2.4 Experimental setup

To ensure a fair comparison between the modified C-Bet and the original C-Bet algorithm, the experiments closely follow the structure proposed by Parisi et al.. They analyzed two sets of results, evaluating the two policies. For the exploration policy, they presented three setups, but this research focuses on two due to specific research objectives. Moreover, due to limited computational resources, the number of environments used at transfer is restricted to four instead of ten. The two setups are:

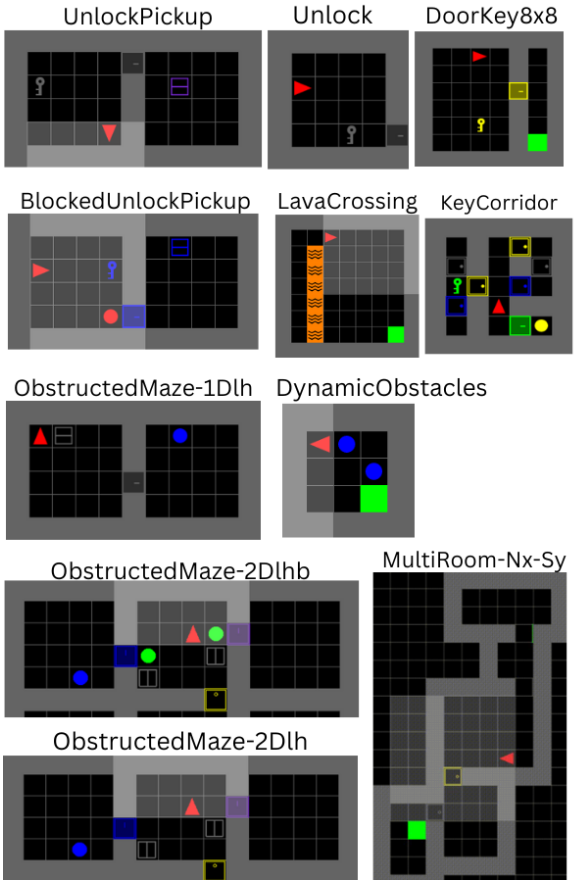

Figure 2.1: The MiniGrid Environments Used in the Experiments. In most environments, the agent has to interact with the objects in order to solve the task. In LavaCrossing and DynamicObstacles, the agent should avoid the objects in the environment, as interacting with these objects would result in termination.

- MultiEnv: The agent is pre-trained on three environments, namely KeyCorridorS3R3, BlockedUnlockPickup, and MultiRoom-N4-S5, sharing one policy across the environments.
- SingleEnv: The agent is pre-trained on a single environment, DoorKey8x8 or KeyCorridorS3R3. The exploration policy is learned separately for each environment.

In each setup, the agent is pre-trained over 25 million frames. It is noteworthy that the original experiments used 50 million frames for pre-training. Due to limited computational resources, this study employs only 25 million frames, which may impact the overall performance of the agents. To determine whether the pre-training policy improves the exploration of the task-specific policy, they transferred it to three new environments: Unlock, UnlockPickup, and ObstructedMaze1Dlh. At transfer, the agent is trained using extrinsic rewards only, while the exploration policy acts as a fixed bias to promote interaction. The duration of training varies depending on the specific transfer environment and it is illustrated in 2.1.

Environment	**Frames**
Unlock	4 million
UnlockPickup	20 million
ObstructedMaze1Dlh	20 million

Table 2.1: Duration of Extrinsic Rewards Training (in frames) for Different Environments

The policies are evaluated by the following criteria:

- Unique interactions: Actions that result in novel environment changes, such as picks, drops, or toggles. Interactions are computed by converting the environment state to its string representation and comparing the current string with the previous one. A change in the state representation indicates an interaction that alters the environment. To identify unique interactions, a dictionary is employed to log all observed changes. Following the pretraining phase, the agent's exploration policy is applied during transfer to monitor unique interactions over 100 episodes across twelve environments. This evaluation includes ten environments from the original study and introduces two new environments, Crossing and Dynamic Obstacles, to assess the impact of the intrinsic reward formulation in scenarios where rare environmental changes may have adverse effects.
- Task success rate: The number of episodes in which the exploration policy reaches goal states, indicating task completion. This metric is also recorded over 100 episodes at transfer to twelve environments, after pretraing.
- Extrinsic return: recorded during training with extrinsic rewards only.

3 Results

3.1 Pre-training results

Figures 3.1 and 3.2 depict the results after pretraining in the MultiEnv setup. We choose to focus on this setup as the Modified algorithm performed best here. The results from the SingleEnv setup are reported in Appendix B . Modified C-Bet interacts less than C-Bet and Count in all environments, as shown in Figure 3.1. Moreover, none of the algorithms interact in the LavaCrossingS9N2 environment.

Environment	**Modified C-Bet**	**C-Bet**	**Count**
Unlock-v0	Higher	Lower	Highest
UnlockPickup	Higher	Lower	Highest
BlockedUnlockPickup	Higher	Lower	Highest
KeyCorridorS3R3	Lower	Higher	Highest
ObstructedMaze-1Dlh	Higher	Lowest	Lower
LavaCrossingS9N2-v0	Low	Low	Low
Dynamic-Obstacles-6x6	Higher	Lower	Higher

Table 3.1: Task Success Rates Comparison between Different Algorithms in Specific Environments

Despite having fewer unique interactions, Modified C-Bet demonstrates a competitive performance with regard to the task success rate, surpassing C-Bet and occasionally Count in several cases, as shown in Figure 3.2. Table 3.1 provides a comparison of task success rates among the algorithms, highlighting the environments where Modified C-Bet performs well. This algorithm has a higher task success rate than C-Bet in five environments. However, Count outperforms Modified C-Bet in three

of these environments, namely Unlock-v0, UnlockPickup, and BlockedUnlockPickup. Furthermore, Modified C-Bet shows a low task success rate compared to the other algorithms in KeyCorridorS3R3, which is one of the environments used during pretrain. Finally, Modified C-Bet outperforms both C-Bet and Count in the ObstructedMaze-1Dlh when it comes to task success rate. As for the two additional environments, LavaCrossingS9N2-v0 and Dynamic-Obstacles-6x6, all algorithms have a relatively low success rate, especially in LavaCrossingS9N2. Count and Modified C-Bet have a comparable performance in terms of success rate in the Dynamic-Obstacles environment, outperforming C-Bet.

3.2 Transfer results

The policies learned during pre-training are transferred to three environments, as described in Section 2.4. Figure 3.3 shows the moving average of the extrinsic return at training using the task-specific policy with the exploration policy as a bias. Firstly, the results in the Unlock environment show that the Modified algorithm has a faster convergence than C-Bet in the Multienv setup and a comparable performance in the Corridor setup. However, in the Doorkey setup, the Modified algorithm has a slower convergence than both C-Bet and Count. Moreover, all three algorithms have a chaotic performance in the UnlockPickup and ObstructedMaze-1Dlh environments, as the lines fluctuate irregularly. Although these results are unstable, it can be observed that the Modified algorithm is slower to converge than C-Bet and Count in four out of the six setups. Only in the Multienv setup of the ObstructedMaze environment does Modified C-Bet outperform C-Bet in terms of speed of convergence, as it is illustrated in Figure 3.3.

4 Discussion

The pre-training results were designed to evaluate the exploration policy learned during pre-training with intrinsic rewards. The transfer results aim to determine whether the exploration policy improves the task-specific policy at transfer to new environments. We will discuss the results obtained in Section 3.

4.1 Pre-training results

Modified C-Bet showed a higher task success rate than C-Bet in five environments, suggesting an improvement over the original algorithm. Moreover, in three of these environments, namely Unlock, UnlockPickup and BlockedUnlockPickup, Modified C-Bet's performance is closer to Count's performance rather than C-Bet's. This observation hints to a potential similarity between Modified C-Bet and Count. To explore this idea further, we analyze the intrinsic reward defined in 2.2. In this formulation, the state counts ($N(s')$) increase faster than the change counts ($N(c)$). For every action, the state changes, leading to an increase in the state count. However, not all actions lead to a change in the environment (e.g. forward). Unlike C-Bet, which resets both counts, Modified C-Bet omits resets, resulting in $N(s') >> N(c)$ over time, such that the change counts become irrelevant. Consequently, in the long run, Modified C-Bet behaves similarly to Count.

Count prioritizes state convergence in familiar environments, as noted by (Parisi et al., 2021), leading to high success rates in environments like Unlock, UnlockPickup, and BlockedUnlockPickup but struggles in unfamiliar environments such as ObstructedMaze-2Dlh and ObstructedMaze-2Dlhb.The Modified C-Bet algorithm shows a resemblance to Count in the Unlock, UnlockPickup and BlockedUnlockPickup environments, having low interactions and a high task success rate. However, in environments like ObstructedMaze-2Dlh and ObstructedMaze-2Dlhb, Modified C-Bet diverges from Count.

In this scenario, Modified C-Bet shows results that differ from Count, having low interactions and a low success rate. These differences can be attributed to the initial influence of change counts in the intrinsic reward. Additionally, Modified C-Bet's performance differs from Count in KeyCorridorS3R3 and ObstructedMaze-1Dlh. Although KeyCorridor is among the environments used for pre-training, Modified C-Bet shows a notably lower task success rate compared to both C-Bet and Count. This disparity may stem from KeyCorridor's complex structure with numerous objects relative to the state space. In such an environment, the change counts are more important, and their diminished impact on the reward may undermine

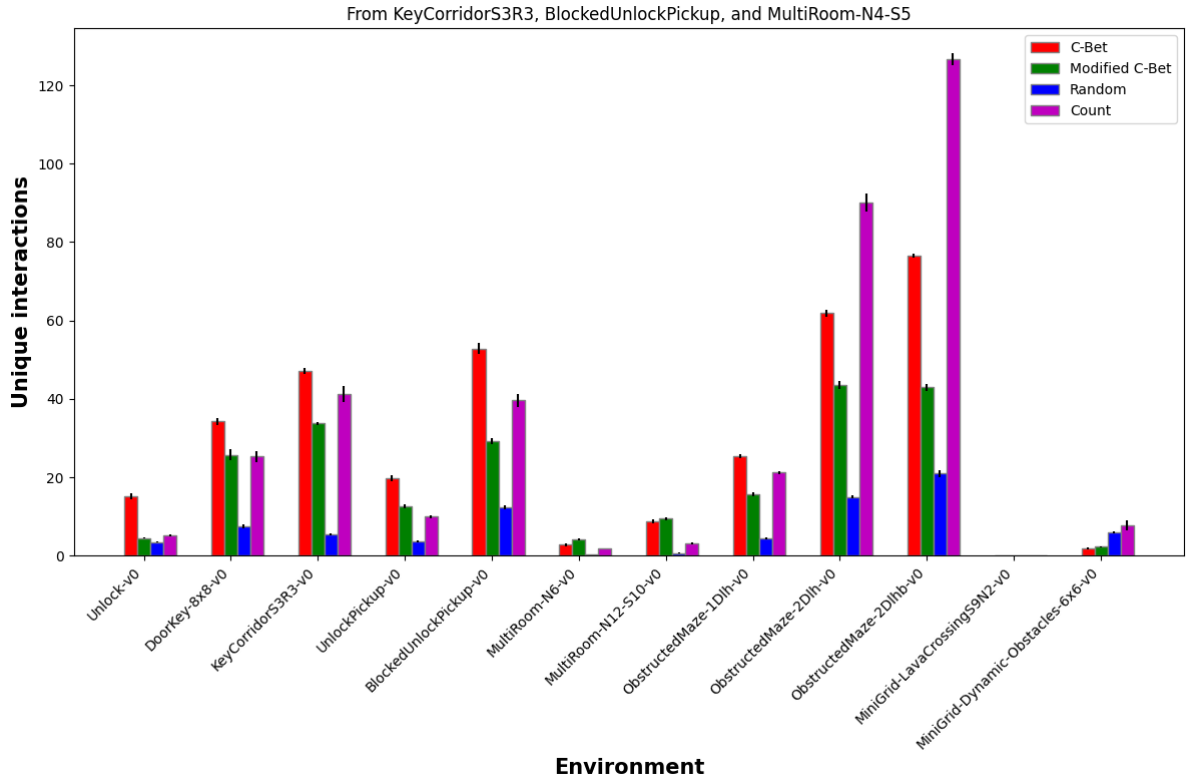

Figure 3.1: Unique Interactions at the Beginning of Transfer to 12 Environments, after Pre-training in Multienv using C-Bet, Count, and Modified C-Bet. At transfer, a random agent was tested as well.

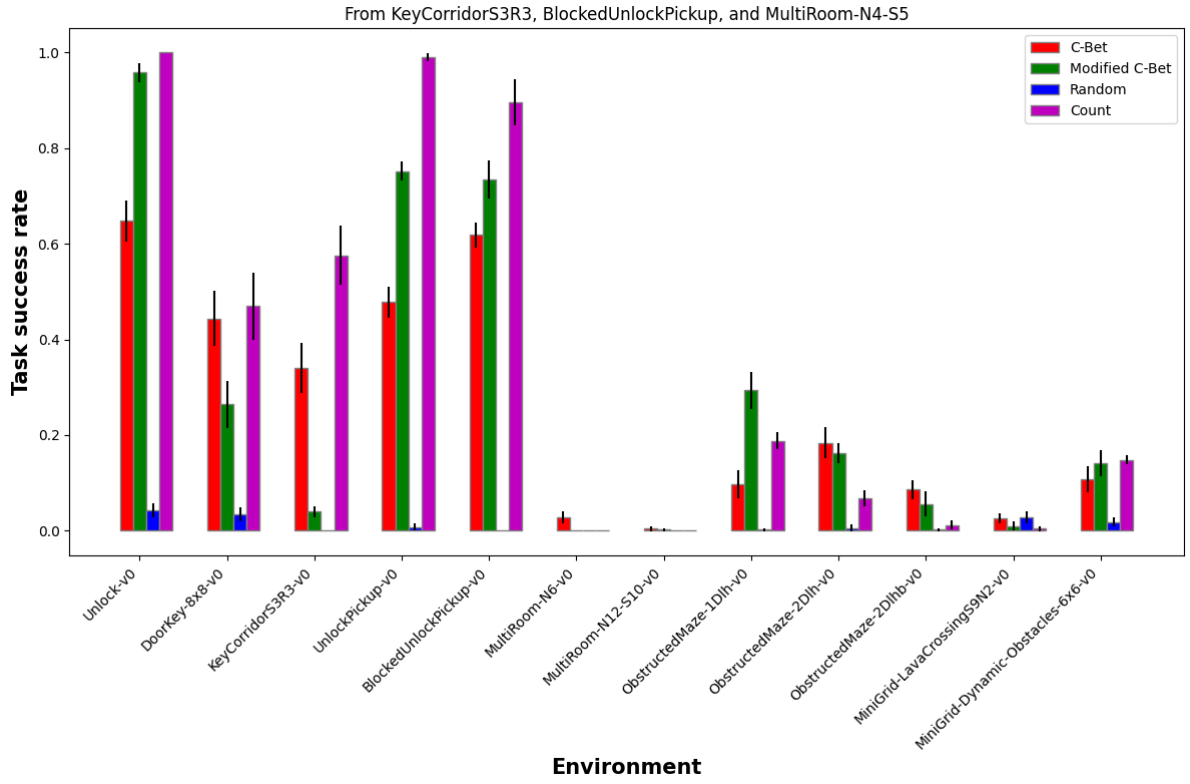

Figure 3.2: Task Success Rate at the Beginning of Transfer to 12 Environments, after Pre-training in Multienv using C-Bet, Count, and Modified C-Bet. At transfer, a random agent was tested as well.

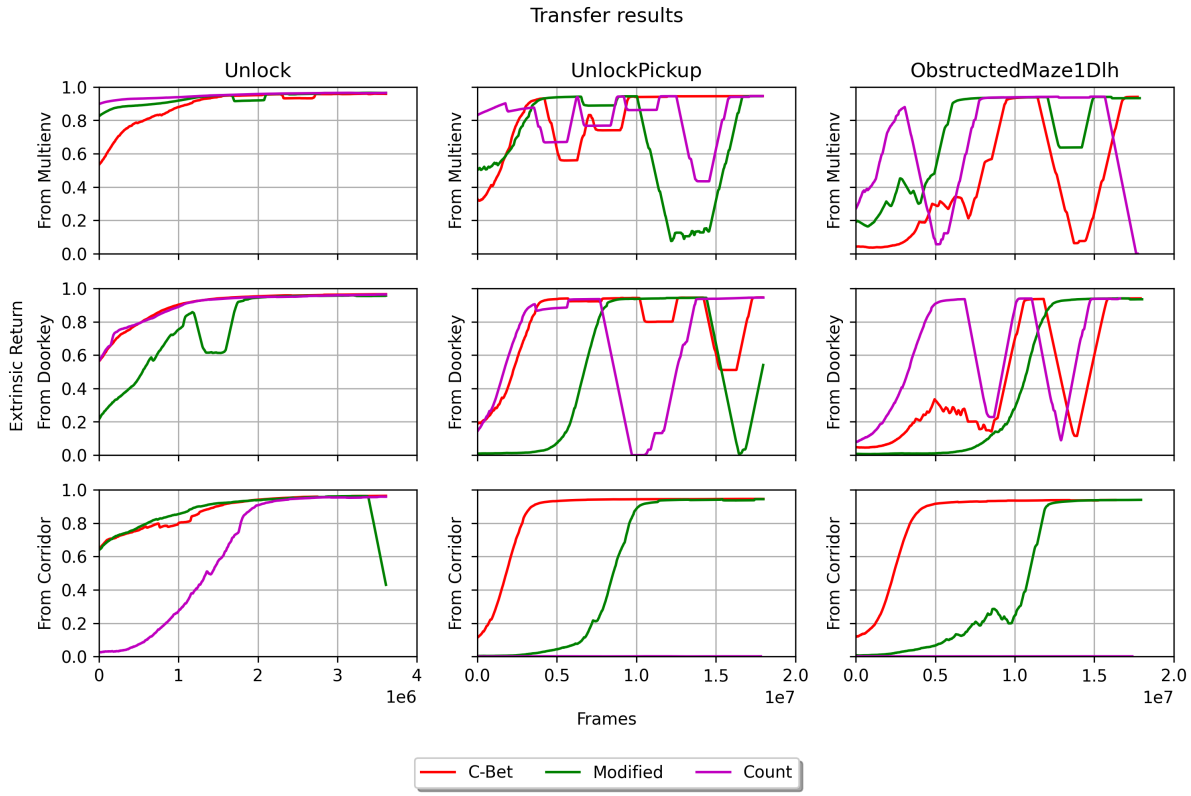

Figure 3.3: Extrinsic Return during Task-specific Learning Using Transferred Policies from three Setups: Multienv, Doorkey, and Corridor. The subplots show the moving average of the extrinsic return for three algorithms: C-Bet, Count, and the Modified algorithm.

Modified C-Bet's performance. Conversely, in environments with a lower ratio of objects to states, such as ObstructedMaze-1Dlh, Modified C-Bet outperforms both C-Bet and Count, indicating that the initial impact of change counts can be advantageous.

Furthermore, introducing new environments like LavaCrossing and Dynamic-Obstacles reveals interesting insights. In LavaCrossing, there are no unique interactions for any algorithm due to the way these interactions are defined and the environment structure. This environment consists of six lava tiles, which terminate the episode in failure, and a goal state. Therefore the agent cannot pick up, drop, or toggle any objects, which would be the only possible way to change the environment and have a unique interaction. The task success rate in LavaCrossing is low for all algorithms, making it inconclusive whether change counts in the intrinsic reward are detrimental. In Dynamic-Obstacles, where the optimal strategy involves minimal object manipulation to avoid interactions, Modified C-Bet aligns closely with this behavior, exhibiting low interactions and high task success rates. This behavior contrasts with C-Bet's lower task success rate, suggesting that actively changing the environment might not be advantageous in Dynamic-Obstacles.

4.2 Transfer results

In the Transfer phase, particularly in the Unlock environment within the Multienv setup, Modified C-Bet has faster convergence compared to C-Bet. This trend corresponds with their respective performance in task success rates as described in the pre-training results. Similar observations hold for other setups, specifically, DoorKey and KeyCorridor, detailed in Figures B.3 and B.4 in Appendix B. These findings suggest that Modified C-Bet's exploration policy enhances extrinsic return during transfer in environments where it previously exhibited high task success rates.

However, the results in environments like UnlockPickup and ObstructedMaze exhibit erratic performance across all algorithms. Fluctuations observed in the Multienv and DoorKey setups indicate instances of policy collapse, characterized by the worsening of the agent's performance as it continues to interact with the environment (Dohare et al., 2023). The agents seem to suffer from catastrophic forgetting, as they forget the good policy they learned, which is shown by the sudden decrease in the extrinsic return. Despite these setbacks, the agents retain their plasticity, demonstrating the ability to re-learn optimal policies after performance dips. Various factors, including exploration policy duration, model architecture such as Impala's off-policy learning, and environment stochasticity, may contribute to this behavior. Despite the instability, Modified C-Bet shows slower convergence than C-Bet and Count in DoorKey and KeyCorridor setups in these environments. However, it outperforms C-Bet in scenarios where its exploration policy previously led to higher task success rates, such as the Multienv setup of ObstructedMaze.

5 Conclusions

This research aimed to determine whether modifying the C-Bet algorithm by adding a time-variant component to the intrinsic reward would improve the algorithm's performance in terms of exploration. Modified C-Bet showed improved performance over C-Bet in several environments during pre-training, particularly aligning with the Count algorithm's behavior. It performed well in environments with fewer objects and larger state spaces but struggled in visually rich environments. During the transfer phase, Modified C-Bet demonstrated faster convergence in setups where it had a high task success rate during pre-training, but exhibited instability in others, likely due to policy collapse and catastrophic forgetting.

Limitations A few limitations are related to how the experiments were set up. Due to the computational complexity of the algorithm, we completed only a single run for the transfer results. Multiple runs could have provided more stable results and a better analysis of overall behavior. Additionaly, the exploration policies were pre-trained for less than described in the original paper. Increasing the number of frames used at pre-training may have improved the exploration policy of the Modified algorithm. Another limitation concerns the generalization abilities of the Modified algorithm. The diminishing impact of the change counts over time appears to have influenced the algorithms' success.

Consequently, its performance is highly dependent on the structure of the environments, having a lower success rate in visually rich environments and a higher success rate in environments with a large state space and few objects. Therefore, Modified C-Bet does not generalize effectively, which may result in poor adaptability to new environments.

Future research One idea for future research is to ensure that the change counts remain a relevant component in the intrinsic reward over a longer time. Assuming that, in the limit, the counts will increase almost linearly, but at different rates, there may be a parameter that would scale these counts accordingly. The intrinsic reward could be reformulated as such:

$$\tilde{r_i}(s, a, s') = \frac{1 + ln(t)}{\rho N(s') + N(c)(1 - \rho)}, \qquad (5.1)$$

where $\rho \subseteq (0, 1)$. This adjustment would scale down state counts while scaling up change counts, maintaining their importance over time. Finding this hyperparameter may improve the performance of Modified C-Bet, especially in environments with a high proportion of objects to states, and enhance its generalization and overall applicability. However, we acknowledge that adding such a parameter will only delay the point at which state counts overtake change counts; ultimately, change counts will still become irrelevant. Nonetheless, this may be beneficial, as the similarity between Modified C-Bet and Count showed that using only state counts may sometimes be an advantage.

Another proposal for future investigations involves the time-variant component of the intrinsic reward. Instead of relying on mathematical intuition to choose this function, one may first analyze the shape of the intrinsic reward over training and use this information to formulate an appropriate function.

In conclusion, the Modified C-Bet algorithm introduces valuable enhancements to the intrinsic reward mechanism, resulting in improved exploration in specific contexts. However, its limitations highlight the need for further refinement to achieve consistent performance across diverse environments. Nevertheless, introducing time functionality as a way to navigate vanishing intrinsic rewards may be worth exploring in the context of sparse rewards reinforcement learning.

References

Barto, A. G. (2012). Intrinsic motivation and reinforcement learning. *Intrinsically motivated learning in natural and artificial systems*, pages 17–47.

Bellemare, M., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., and Munos, R. (2016). Unifying count-based exploration and intrinsic motivation. *Advances in neural information processing systems*, 29.

Chevalier-Boisvert, M., Dai, B., Towers, M., de Lazcano, R., Willems, L., Lahlou, S., Pal, S., Castro, P. S., and Terry, J. (2023). Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. *CoRR*, abs/2306.13831.

Dohare, S., Lan, Q., and Mahmood, A. R. (2023). Overcoming policy collapse in deep reinforcement learning. In *Sixteenth European Workshop on Reinforcement Learning*.

Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., et al. (2018). Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International conference on machine learning*, pages 1407–1416. PMLR.

Nahirnyi, O. (2022). Reinforcement learning agents in procedurally-generated environments with sparse rewards.

Parisi, S., Dean, V., Pathak, D., and Gupta, A. (2021). Interesting object, curious agent: Learning task-agnostic exploration. *Advances in Neural Information Processing Systems*, 34:20516–20530.

Raileanu, R. and Rocktäschel, T. (2020). Ride: Rewarding impact-driven exploration for procedurally-generated environments. *arXiv preprint arXiv:2002.12292*.

Ryan, R. M. and Deci, E. L. (2000). Intrinsic and extrinsic motivations: Classic definitions and new directions. *Contemporary educational psychology*, 25(1):54–67.

Schmidhuber, J. (1991). A possibility for implementing curiosity and boredom in model-building neural controllers.

Sutton, R. S. (1991). Reinforcement learning architectures for animats.

Tang, H., Houthooft, R., Foote, D., Stooke, A., Xi Chen, O., Duan, Y., Schulman, J., DeTurck, F., and Abbeel, P. (2017). # exploration: A study of count-based exploration for deep reinforcement learning. *Advances in neural information processing systems*, 30.

A Hyperparameter details: Intrinsic reward coefficient

To find an appropriate coefficient for the modified algorithm, We looked at the intrinsic reward value of both C-Bet and the modified version after 12 million frames at pre-training in KeyCorridorS3R3. First, we estimated the average intrinsic reward after 12 million frames for both algorithms. Then we calculated the multiplicative difference and used it for a factor, leading to a value of 0.00625.

Figure A.1: Moving Average of the Intrinsic Rewards achieved by C-Bet and the Modified C-Bet at Pre-training over 12 million Frames in the Environment KeyCorridorS3R3.

B Extended results

Figures B.2, B.1, B.4, B.3 illustrate the unique interactions and task success rate from the two SingleEnv setups, DoorKey and KeyCorridor. Modified C-Bet consistently exhibits fewer unique interactions compared to C-Bet across all environments in both setups. Regarding task success rates in the KeyCorridor setup, Modified C-Bet outperforms both C-Bet and Count in four environments. The lower performance observed in these setups underscores the limitation of pre-training in a single environment.

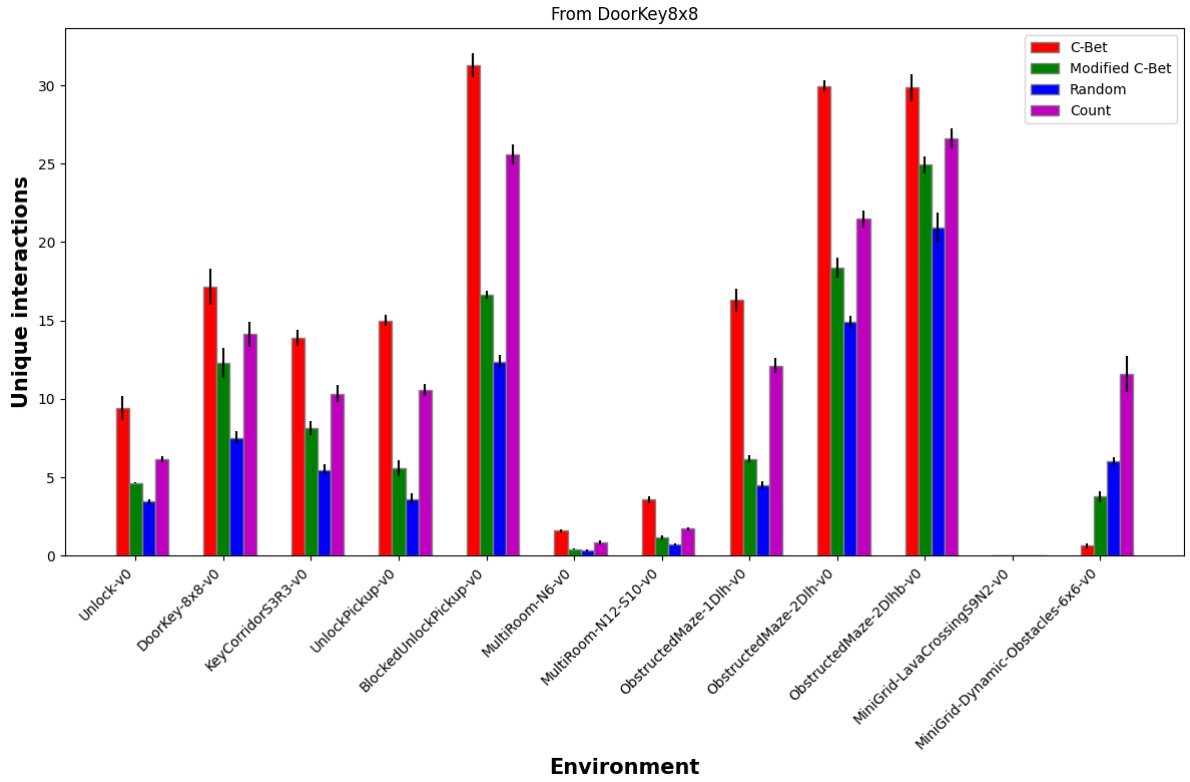

Figure B.1: Unique Interactions at the Beginning of Transfer to 12 Environments, after Pre-training in DoorKey8x8 using C-Bet, Count, and Modified C-Bet.

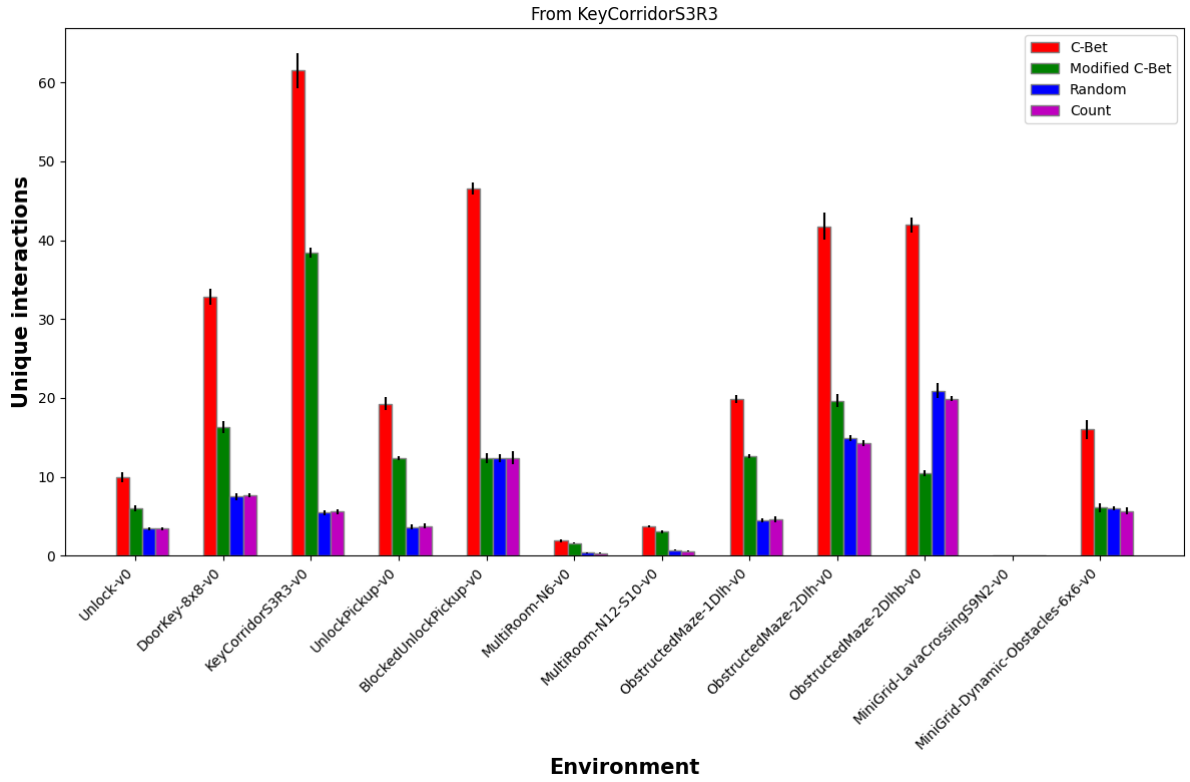

Figure B.2: Unique Interactions at the Beginning of Transfer to 12 Environments, after Pre-training in KeyCorridorS3R3 using C-Bet, Count, and Modified C-Bet.

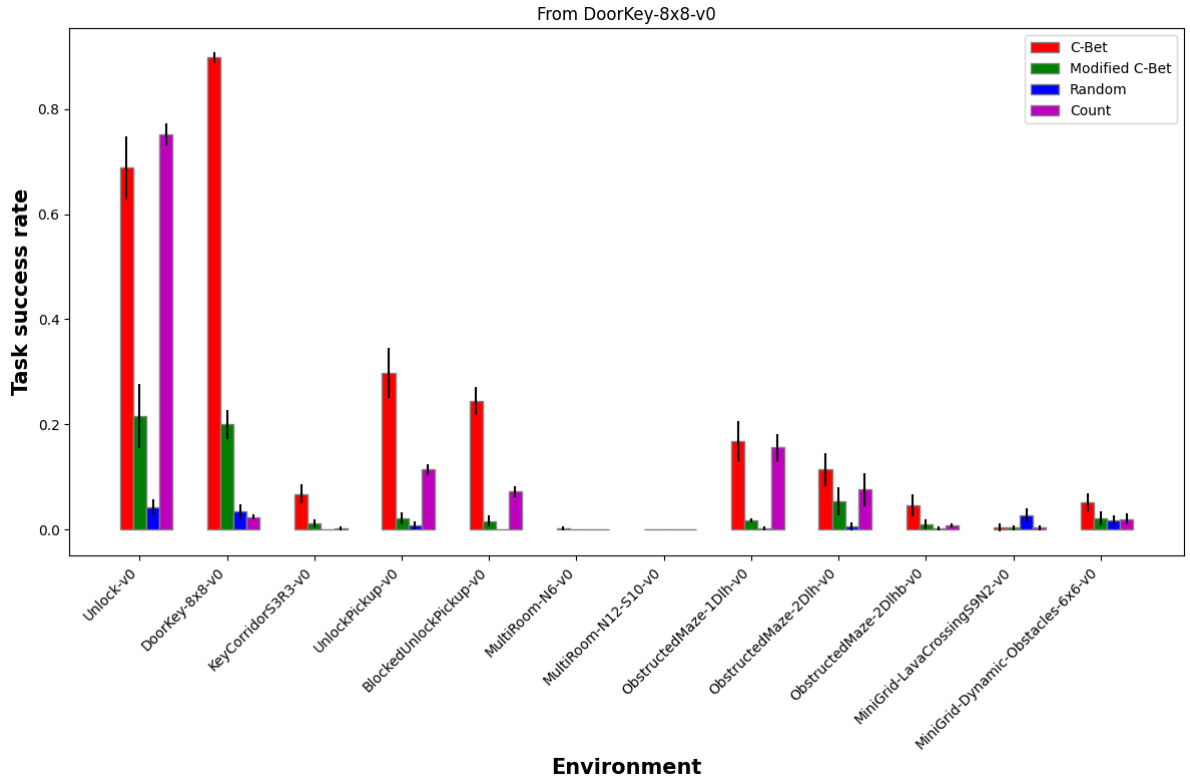

Figure B.3: Task Success Rate at the Beginning of Transfer to 12 Environments, after Pre-training in DoorKey8x8 using C-Bet, Count, and Modified C-Bet.

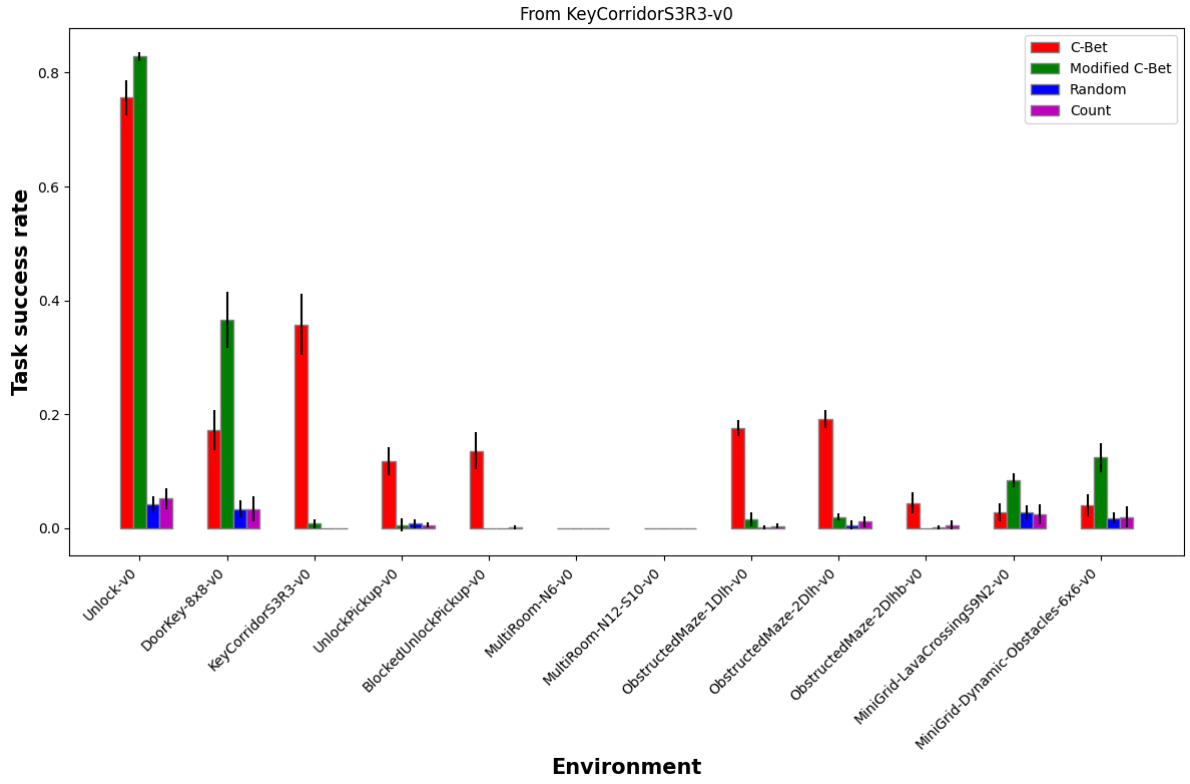

Figure B.4: Task Success Rate at the Beginning of Transfer to 12 Environments, after Pre-training in KeyCorridorS3R3 using C-Bet, Count, and Modified C-Bet.